A Artifact Appendix

A.1 Abstract

Our artifacts demonstrate how our hardware prefetcher attacks FetchProbe and ShadowLoad work. The artifacts include our tool StrideRE and proof-of-concepts from the paper and allow reproduction of their results. The artifacts require a 12th or 13th-generation Intel processor and an Intel processor affected by Meltdown (e.g., Skylake or Coffee Lake). The artifacts only work on x86_64 Linux and require root access.

A.2 Artifact check-list (meta-information)

- **Hardware:** Intel 12th Gen or 13th Gen processor + Intel processor affected by Meltdown (e.g., Skylake or Coffee Lake)
- **Run-time environment:** Linux, Root required
- **How much disk space required (approximately)?:** 1GB
- **How much time is needed to prepare workflow (approximately)?:** a few minutes
- **Experiments:** Python scripts for evaluation are available
- **How much time is needed to complete experiments (approximately)?:** about 12 hours to several days (if Collide + Power is evaluated)
- **Publicly available?:** yes
- **Code licenses (if publicly available)?:** MIT / GPL
- **Archived (provide DOI)?:** 10.6084/m9.figshare.28381319

A.3 Description

A.3.1 How to access.
The artifacts can be obtained in the following repository: github.com/cispa/ShadowLoad.

A.3.2 Hardware dependencies.

- 12th or 13th-generation Intel processor
- Intel processor affected by Meltdown (e.g., Skylake or Coffee Lake)

We tested the provided artifacts using an Intel Core i7-8700k and an Intel Core i9-12900k. Two processors are required as our provided attacks are fine-tuned for hardware prefetchers that can cross page boundaries, and other attacks require processors vulnerable to Meltdown. The former is only available on newer processors, while the latter is only available on old Intel processors. Some of the provided artifacts can be executed with different processors but might require additional tweaking.

A.3.3 Software dependencies.
Linux operating system (no VM), git, kernel headers, GNU Make, gcc, python3, matplotlib. We used Ubuntu 22.04 as an operating system during testing. Matplotlib should also be set up work when python3 is executed as root.

A.4 Installation

After downloading the artifacts, run the *compile_and_check.py* as root (*sudo python3 compile_and_check.py*). This script compiles all examples and kernel modules and should terminate with exit code 0 or provide more information on failure.

A.5 Experiment workflow

Experiments are divided into different directories. Most directories contain an *eval.py* script that will automatically run the experiment and reports the results or stores them in a directory called *out*. The expected results and more details on how each experiment can be run are detailed in the next section.

A.6 Evaluation and expected results

00_fetch_probe. (5-10 minutes)
The experiment should be executed on a 12th or 13th-generation Intel processor. The *eval.py* script must be executed as root. An example output contains the following lines:

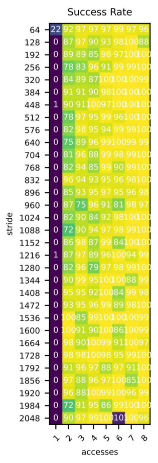
```
(...)
times: 0.022398622
correct: 32767.0
false_positives: 0.0
false_negatives: 1.0
positives: 16383.5
negatives: 16384.5
precision: 100.0%
recall: 100.0%
f-score: 100.0%

(...)
times: 0.043121079
correct: 32766.0
false_positives: 0.0
false_negatives: 2.0
positives: 16380.0
negatives: 16388.0
inv_correct: 32766.0
inv_false_positives: 0.0
inv_false_negatives: 2.0
inv_positives: 16388.0
inv_negatives: 16380.0
precision: 100.0%
recall: 100.0%
f-score: 100.0%
inv_ precision: 100.0%
inv_ recall: 100.0%
inv_ f-score: 100.0%
```

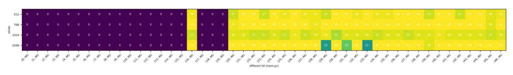
The experiment is successful if the amount of false positives and false negatives is low. The precision, recall, and f-score should be comparable to the numbers shown in the paper but at least 80%.

01_shadow_load. (about 5 minutes)
The experiment should be executed on a 12th or 13th-generation Intel processor. The *eval.py* script must be executed as root. After execution, the *out* directory should contain, amongst others, the files *result_shadowload_1.svg*

and *result_shadowload_kernel_1.svg*. These images should show cache hits starting with two accesses:



02_stride_re. (up to 12 hours)
We recommend using a 12th or 13th-generation Intel processor for this experiment. The *eval.py* script must be executed as root. This experiment may take up to 12 hours to complete. After execution, the *tests/out* directory should contain many files. For instance, the file *test_prefetch_both_collisions_mem_aligned_2_rdtsc,kernel,-DEVAL,-DACCESS_MEMORY.svg* which visualizes the relevant bits in the accessed address for userspace to kernel mistraining of the hardware stride prefetcher with two training accesses and an aligned trigger access in the kernel. This should look comparable to this example on 12th or 13th-generation Intel processors:



03_base64. (2 - 10 seconds)
The experiment should be executed on a 12th or 13th-generation Intel processor. The *eval.py* script can be executed as an unprivileged user. The experiment is successful if the incorrect rate is low (it should be below 30%). The incorrect rate describes the number of runs out of 1000, where at least one bit was incorrect.

Sample output:

```
(...)
leakage: 87104.25 bit/sec.
incorrect: 194 / 1000 (19.40%
unknown left: 16.56575682382134
invokations : 1238.1042183622828
```

04_meltdown. (30-90 minutes)
The experiment should be executed on an Intel processor vulnerable to Meltdown. Page-table-isolation should be disabled (*mitigations=off* or *nopti* kernel parameter).
On Ubuntu 22.04, the kernel parameters can be changed by editing the *GRUB_CMDLINE_LINUX_DEFAULT* configuration in */etc/default/grub* and executing *update-grub* afterward. The *analyze.py* script (run by the *eval.py* script) further enables huge pages using `sysctl -w vm.nr_hugepages=50`. The script must be modified if this command is does not apply to the system. The *eval.py* script must be executed as root. The amount of correct bytes should be at least 50% (50% is better than random guessing which would be $\frac{1}{255}$).

05_spectre. (approximately 3 minutes)
The experiment should be executed on a 12th or 13th-generation Intel processor. The *eval.py* script must be executed as root. The experiment is successful if the amount of correct bytes is high (should be at least 80%).

Sample output:

```
(...)
rate   : 18.2KB/s
correct: 99.4%
--------------------
(...)
```

06_collide_power. (several days)
We included the Collide+Power victim for completeness. It can be evaluated using the open source tool by Kogler et al.: github.com/0xhilbert/rda. Further instructions are included in the *README.md* in the repository at *06_collide_power/runner/user/README.md.*

A.7 Notes

If anything fails, ensure that no kernel modules from other experiments are still loaded.

A.8 Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- https://cTuning.org/ae